|  |  |
| --- | --- |
| **Title** | **Program Automation: Learning from Tests**  **Defined in Unit Testing** |
| **Student name:** | **Rihards Baranovskis** |
| **Supervisor name:** | **Dr R J Walters** |
|  | |
| ***Aims/research question and Objectives*** | |
| Machine learning based program synthesis has gained extra researchers` attention due to rise of neural networks. Most of the current research focuses on program synthesis, which is based on high level - linguistic definitions, for automating the process for the people who are non-programmers. Very little research has been done in regards of implementing such synthesis models in the actual the actual software development process, carried out by IT professionals.  The aim of this project lies in the following research question is: *Can recurrent neural network be used effectively to synthesise software from unit test definitions?* This neural network based system would train from input/output samples, such as those defined in unit testing, as result inducing the desired software. Consequently, the main objectives of the project are:   1. **Familiarisation with recurrent neural Networks based Program Synthesis**   It has been shown that recurrent neural networks can procure highly accurate program synthesis models. Familiarisation with design patterns and subtleties of training such networks is required before attempting building a functioning synthesis model.   1. **Designing a suitable Neural Programmer Architecture**   Appropriate induction model architecture has to be developed for inducing software from unit test definitions. Known ALU and recurrent neural networks based Neural Programmer architectures exist for inducing latent programs form high level linguistic definitions. These will have to be redesigned to fit the research problem.   1. **Training and Tuning, and Testing the model**   The designed synthesis model should be trained and its hyper-parameters tuned based on the test results from pre-generated input/output samples. These will be produced from know functions and their relevant unit tests definitions.   1. **Attempting to design a generalised meta-model**   This is a crucial objective for proving the effectives of such synthesis system. Traditionally, neural networks require big amount of training samples to produce generalized model. Meta model would allow to pre-train a synthesis model, which then can be trained for a particular case with small amount of training data.   1. **Comparing the results with non-machine learning based methods**   Evaluation of the final model performance will require comparison with the synthesis results of the “programming by example” program induction tools. This will allow making an objective judgement on the potential use of the neural programmer for this kind of problem.   1. **Testing the architecture at a lower level.**   If time permits, the neural programmer ALU unit and the input/output samples will be redefined at a lower programming level and then tested to find if that allows to increase the model’s accuracy. Redefining the problem at the assembly level can potentially help to train better generalized model, particularly for non-numerical inputs.   1. **Writing the thesis and reporting on the results.**   Final objective of this project is to produce a thesis paper, which enlists the scope of experimentation done, results analysis, critical discussion and final conclusion. | |

|  |
| --- |
| ***Summary of proposed research and analysis methodology*** |
| The main focus of the research will be using an iterative process of finding the most suitable neural programmer architecture with the best corresponding hyperparmeters for the trained recurrent neural network. The model is going to be built using Tensorflow Machine Learning library[[1]](#footnote-2) and possibly trained on a GPU cluster. It is anticipated to find good tuning and architectural practices by reviewing research and vast amount of experimentation, many of which can be difficult to foresee prior to starting the actual hands-on work. The accuracy of this model will measured used so called Generalization criteria. A scalar answer is considered to be correct if it is equal up to the first decimal place with the expected value. The overall performance of the model is going to be given by a ratio between the amount of correct and all testing samples. The analysis of the final will conducted by comparing it to the non-machine based synthesis tools, such as Microsoft Prose[[2]](#footnote-3). The objective measure of success will be defined as the difference in the test results between the developed model and non-machine learning based models.  The model is going to be trained to induce function of varied difficulty, starting from simple mathematical functions up to scalar output based programming challenges, taken from sources such as codility[[3]](#footnote-4), coderbyte[[4]](#footnote-5) and others. These functions will be used as input/output pairs generators, to synthesise the data for the model. Then data is going to be shuffled, split into test and train pairs. As result for each function training and testing performance is going to be determined.  The program induction model is going to be based on the Neural Programmer architecture, as seen below. The controller unit consists of two recurrent neural networks, one for encoding the input; second one for memorising all the previous steps taken. At each timestep the model selects a simple operator from ALU to apply on the input data, thereby producing a sequence of operations. The selector produces a probability distribution for selection of the operations, as  ,  where and are input and history representation at time step , is the parameter matrix for producing the probability distribution, is the matrix representation of the selectable operations. During the training phase, the action is selected using a softmax action selection rule, for differentiability, and during the testing phase hardmax approach is preferred. Both recurrent nets are trained using the stochastic gradient descent with Adam optimizer to minimise the loss function, which in case of scalar output is defined as a Huber Loss. At each step, operation results and operators selected are saved in the memory, which eases the interpretation of the final output.    Figure 1 - Neural Programmer Architecture Example[[5]](#footnote-6)  During the project executing, good development and research practices are going to be used, as program code commenting and documentation, version control, model training and testing automation, hyperparameters search automation, prior research and evidence based experimentation. |

|  |
| --- |
| ***Research plan – Gantt chart or Pert chart*** |
|  |

|  |
| --- |
| ***Ethical statement*** |
| **AI – leaving the developers out of work**  **“This week, scientists and economic advisers to President Obama released a report on artificial intelligence, including the effects of automation on the US job market and economy. While the report notes the significant potential for wealth gains from increased productivity due to AI, it also warns of threats to existing jobs and an exacerbation of the wage inequality between lower-skilled, less-educated workers and those with higher skills.”**  [**https://arstechnica.co.uk/business/2016/12/federal-report-ai-could-threaten-up-to-47-percent-of-jobs-in-two-decades/**](https://arstechnica.co.uk/business/2016/12/federal-report-ai-could-threaten-up-to-47-percent-of-jobs-in-two-decades/)  **“McDonald’s ex-CEO: $15/hr minimum wage will unleash the robot rebellion” – wage gap, inequlity will only be enpowered by the AI and automation progress, as business will choose to deal with machines and algorithms rather than demandful employees. As due to low wages employess already exort pressure on people.**  [**https://arstechnica.com/business/2016/05/mcdonalds-ex-ceo-15hr-minimum-wage-will-unleash-the-robot-rebellion/**](https://arstechnica.com/business/2016/05/mcdonalds-ex-ceo-15hr-minimum-wage-will-unleash-the-robot-rebellion/)  **Plagiaritism**    **Something bad for another people happening ?** |

|  |
| --- |
| ***Legal and commercial aspects*** |
| **Meta models – can possibly**  **Legal – developing the model with open source/uni software – then using it in a commercial project.**  **Does uni own any of the work you do? – yes, all of it.** |

1. [*https://www.tensorflow.org/tutorials/recurrent*](https://www.tensorflow.org/tutorials/recurrent) [↑](#footnote-ref-2)
2. [*https://microsoft.github.io/prose/*](https://microsoft.github.io/prose/) [↑](#footnote-ref-3)
3. [*https://codility.com/programmers/challenges/*](https://codility.com/programmers/challenges/) [↑](#footnote-ref-4)
4. [*https://coderbyte.com/challenges*](https://coderbyte.com/challenges) [↑](#footnote-ref-5)
5. *Q.V. and Sutskever, I. Le, "Neural Programmer: Inducing Latent Programs with Gradient Descent," arXiv:1511.04834, 2016* [↑](#footnote-ref-6)